POLITEKNIK NEGERI BANYUWANGI

Laporan Final Project: Smart E-Kantin

Mobile Application Development
Semester Ganjil 2024/2025 [cite: 4]

Disusun Oleh Kelompok 04:

Danish Naisyila Azka	362458302098	*Backend Architect*
Dian Restu Khoirunnisa	362458302094	*UI Engineer*
Vina Faizatus Sofita	362458302095	*Auth & Navigation*
Nadhifah Afiyah Qurota'ain	362458302100	*Transaction Logic*

Tahun 2025

BAB 1
Pendahuluan & Pembagian Kerja

1.1 Deskripsi Aplikasi

Aplikasi CineBooking adalah sistem pemesanan tiket bioskop berbasis mobile yang dibangun menggunakan Flutter. Aplikasi ini memungkinkan pengguna untuk melihat daftar film, memilih kursi, dan melakukan booking secara real-time dengan integrasi Firebase sebagai backend.

Fitur utama aplikasi:
- Autentikasi pengguna (Login & Register)
- Daftar film dengan informasi detail
- Pemilihan kursi interaktif
- Sistem booking dengan QR Code
- Riwayat transaksi pengguna
- Perhitungan harga otomatis dengan diskon kursi genap

1.2 Teknologi yang Digunakan

- **Framework**: Flutter
- **Backend**: Firebase (Authentication, Firestore)
- **State Management**: Provider
- **Database**: Cloud Firestore
- **Version Control**: Git

1.3 Pembagian Tugas Tim

Nama	NIM	Peran
Danish Naisyila Azka	362458302098	Backend Architect
Dian Restu Khoirunnisa	362458302094	UI Engineer
Vina Faizatus Sofita	362458302094	Auth & Navigation
Nadhifah Afiyah Qurota'ain	362458302100	Transaction Logic

1.3.1 Detail Tanggung Jawab

Azka (Backend Architect):
- Setup Firebase dan konfigurasi project
- Membuat model data (MovieModel, UserModel, BookingModel)
- Implementasi FirebaseService untuk CRUD operations
- Integrasi database Firestore

Dian (UI Engineer):
- Desain dan implementasi HomeScreen
- Membuat MovieCard widget
- Implementasi MovieDetailScreen
- Styling dan theming aplikasi

Vina (Auth & Navigation):

- Implementasi LoginScreen dan RegisterScreen
- Handling autentikasi Firebase
- Setup routing dan navigation
- Membuat SeatSelectionScreen

Nadif (Transaction Logic):
- Implementasi BookingController dengan Provider
- Logika pemilihan kursi dan validasi
- Sistem perhitungan harga dan diskon
- ProfileScreen dengan QR Code booking
- CalculationService untuk business logic

BAB 2
Bukti Keaslian Kode

2.1 Watermark Code (Suffix Naming)

Untuk membuktikan keaslian kode dan mencegah penggunaan code generator, setiap anggota tim menggunakan suffix nama pada class, function, dan variable penting.

2.1.1 Contoh Implementasi Suffix

Azka (Backend):

```
class FirebaseServiceAzka {
  final FirebaseAuth _auth = FirebaseAuth.instance;
  final FirebaseFirestore _db = FirebaseFirestore.instance;

  Future<User?> registerUser(...) async { }
  Future<List<MovieModelAzka>> getMovies() async { }
}

class MovieModelAzka {
  final String movieId;
  final String title;
  // ...
}

class UserModelAzka {
  final String uid;
  final String email;
  // ...
}

class BookingModelAzka {
  final String bookingId;
  final String userId;
  // ...
}
```

Dian (UI):

```
class HomeScreenDian extends StatefulWidget { }

class MovieCardDian extends StatelessWidget {
  final dynamic movie;
  final VoidCallback onTap;
  // ...
}

class MovieDetailScreenDian extends StatefulWidget {
  final String movieId;
  // ...
}
```

Vina (Auth & Seat):

```
class SeatSelectionScreenVina extends StatefulWidget {
  final String movieId;
  final String movieTitle;
  final int basePrice;
  // ...
}

class SeatItemVina extends StatelessWidget {
  final String seatId;
  final bool isSold;
  final bool isSelected;
  // ...
}
```

Nadif (Logic):

```
class BookingControllerNadif extends ChangeNotifier {
  List<String> _selectedSeats = [];
  List<BookingModelAzka> _userBookings = [];
  // ...
}

class CalculationServiceNadif {
  static bool validateStudentEmail(String email) { }
  static int calculateTotalPrice(...) { }
  // ...
}

class ProfileScreenNadif extends StatefulWidget {
  final String? highlightBookingId;
  // ...
}
```

2.2 Logic Trap (Business Logic Unik)

Aplikasi ini memiliki logika bisnis khusus untuk perhitungan harga yang tidak umum digunakan di template generator:

2.2.1 1. Diskon Kursi Genap (10%)

Kursi dengan nomor genap mendapatkan diskon 10% dari harga dasar.

```
static int parseSeatNumber(String seatCode) {
  try {
    return int.tryParse(seatCode) ?? 0;
  } catch (_) {
    return 0;
  }
}

static bool isEvenSeat(String seatCode) {
  final seatNumber = parseSeatNumber(seatCode);
  return seatNumber % 2 == 0;
```

```
}

static int calculateSeatPrice(String seatCode, int basePrice) {
  if (isEvenSeat(seatCode)) {
    // Diskon 10% untuk kursi genap
    return (basePrice * (1 - AppConstants.evenSeatDiscount)).toInt();
  }
  return basePrice;
}
```

Contoh Perhitungan:
- Kursi 1 (ganjil): Rp 50.000
- Kursi 2 (genap): Rp 45.000 (diskon 10%)
- Kursi 3 (ganjil): Rp 50.000
- Kursi 4 (genap): Rp 45.000 (diskon 10%)

2.2.2 2. Pajak Judul Film

Jika judul film lebih dari 10 karakter, dikenakan pajak Rp 2.500 per kursi.

```
static int calculateTitleTax(String title, int seatCount) {
  return title.length > 10
    ? AppConstants.titleTax * seatCount
    : 0;
}
```

Contoh:
- "Inception" (9 karakter) → Tidak kena pajak
- "The Dark Knight" (16 karakter) → Pajak Rp 2.500 × jumlah kursi

2.2.3 3. Total Perhitungan Harga

```
static int calculateTotalPrice({
  required List<String> seats,
  required String movieTitle,
  required int basePrice,
}) {
  if (seats.isEmpty) return 0;

  int total = 0;

  // Hitung harga per kursi (dengan diskon genap)
  for (final seat in seats) {
    total += calculateSeatPrice(seat, basePrice);
  }

  // Tambah pajak judul
  total += calculateTitleTax(movieTitle, seats.length);

  return total;
}
```

Contoh Kasus:
- Film: "The Dark Knight Returns" (24 karakter)

- Kursi: [1, 2, 3]
- Base Price: Rp 50.000

Perhitungan:
- Kursi 1 (ganjil): Rp 50.000
- Kursi 2 (genap): Rp 45.000
- Kursi 3 (ganjil): Rp 50.000
- Subtotal: Rp 145.000
- Pajak judul (3 kursi × Rp 2.500): Rp 7.500
- **Total: Rp 152.500**

2.2.4 4. Price Breakdown Display

```
static String generatePriceBreakdown({
  required String movieTitle,
  required List<String> seats,
  required int basePrice,
}) {
  final total = calculateTotalPrice(
    seats: seats,
    movieTitle: movieTitle,
    basePrice: basePrice,
  );

  final evenSeats = seats.where((seat) => isEvenSeat(seat)).length;
  final oddSeats = seats.length - evenSeats;
  final titleTax = calculateTitleTax(movieTitle, seats.length);
  final evenSeatDiscount = (basePrice * AppConstants.evenSeatDiscount *
evenSeats).toInt();
  final hasTitleTax = movieTitle.length > 10;

  return '''
=== PRICE BREAKDOWN ===
Base Price: Rp $basePrice x ${seats.length}
Even Seats ($evenSeats): -10% each (Rp $evenSeatDiscount discount)
Odd Seats ($oddSeats): Regular price
${hasTitleTax
    ? "Title Tax (${movieTitle.length} chars): +Rp $titleTax"
    : "Title Tax: No tax"
}

TOTAL: Rp $total
''';
}
```

2.3 Validasi Email Mahasiswa

Sistem hanya menerima email dengan domain `@student.univ.ac.id`:

```
static bool validateStudentEmail(String email) {
  return email.endsWith(AppConstants.studentEmailSuffix);
}
```

```dart
// Konstanta
class AppConstants {
  static const String studentEmailSuffix = '@student.univ.ac.id';
}
```

Implementasi di LoginScreen:

```dart
if (!_isLogin) {
  if (!CalculationServiceNadif.validateStudentEmail(
    _emailController.text)) {
    setState(() => _error =
      'Only student emails allowed (@student.univ.ac.id)');
    return;
  }
}
```

2.4 QR Code dengan Format Khusus

Setiap booking menghasilkan QR Code dengan format unik yang tidak bisa digenerate otomatis:

```dart
static String generateQRData({
  required String bookingId,
  required String movieTitle,
  required List<String> seats,
  required int totalPrice,
  required DateTime bookingDate,
  required String movieId,
  required String userId,
}) {
  final formatter = DateFormat('dd/MM/yyyy HH:mm');
  return '''
🎬🎟️ CINEBOOKING TICKET 🎬🎟️
==============================
MOVIE: $movieTitle
SEATS: ${seats.join(', ')}
TOTAL: Rp $totalPrice
DATE: ${formatter.format(bookingDate)}
BOOKING ID: $bookingId
USER ID: ${userId.substring(0, 8)}
MOVIE ID: $movieId
==============================
⚠ This ticket is PERMANENT
⚠ Non-refundable & Non-transferable
==============================
SCAN FOR THEATER ENTRY
''';
}
```

Contoh Output QR:

```
🎬🎟️ CINEBOOKING TICKET 🎬🎟️
==============================
MOVIE: Inception
```

```
SEATS: 1, 2, 5
TOTAL: Rp 142500
DATE: 23/12/2025 14:30
BOOKING ID: 7a3b9c4d-2e1f-4a5b-8c6d-9e7f1a2b3c4d
USER ID: abc12345
MOVIE ID: movie_001
==============================
⚠ This ticket is PERMANENT
⚠ Non-refundable & Non-transferable
==============================
SCAN FOR THEATER ENTRY
```

2.5 Seat Summary Helper

```dart
static String getSeatSummary(List<String> seats) {
  if (seats.isEmpty) return 'No seats selected';

  final evenSeats = seats.where((seat) => isEvenSeat(seat)).toList();
  final oddSeats = seats.where((seat) => !isEvenSeat(seat)).toList();

  String summary = '';
  if (evenSeats.isNotEmpty) {
    summary += 'Even seats: ${evenSeats.join(', ')} (-10% each)\n';
  }
  if (oddSeats.isNotEmpty) {
    summary += 'Odd seats: ${oddSeats.join(', ')} (regular price)';
  }
  return summary.trim();
}
```

2.6 Kesimpulan Bukti Keaslian

Dari implementasi di atas, dapat dibuktikan bahwa:

1. **Suffix Naming**: Setiap class menggunakan nama anggota sebagai suffix (Azka, Dian, Vina, Nadif)
2. **Logic Trap Unik**:
   - Diskon kursi genap (tidak umum)
   - Pajak berdasarkan panjang judul
   - Format QR dengan template khusus
3. **Validasi Custom**: Email harus domain mahasiswa
4. **Business Logic Kompleks**: Perhitungan harga dengan multiple factor

Semua ini membuktikan bahwa kode ditulis manual oleh tim, bukan hasil code generator.

BAB 3
Arsitektur Backend

Backend aplikasi CineBooking dikerjakan oleh Azka menggunakan Firebase sebagai Backend-as-a-Service (BaaS).

3.1 Struktur Database Firestore

Aplikasi menggunakan 3 collection utama:

3.1.1 1. Collection: `users`

Menyimpan data pengguna yang terdaftar.

Fields: email, username, created_at, balance, role, status

```
Future<User?> registerUser({
  required String email,
  required String password,
  required String username,
}) async {
  final cred = await _auth.createUserWithEmailAndPassword(
    email: email.trim(), password: password.trim());

  await _db.collection('users').doc(cred.user!.uid).set({
    'email': email.trim(),
    'username': username.trim(),
    'created_at': FieldValue.serverTimestamp(),
    'role': 'user',
    'status': 'active',
  });

  return cred.user;
}
```

3.1.2 2. Collection: `movies`

Menyimpan informasi film yang tersedia.

Fields: title, poster_url, base_price, rating, duration, description, booked_seats

```
class MovieModelAzka {
  final String movieId;
  final String title;
  final int basePrice;
  final double rating;
  final List<String> bookedSeats;

  int get availableSeats => 48 - bookedSeats.length;
  bool isSeatAvailable(String seatId) => !bookedSeats.contains(seatId);
}

Future<List<MovieModelAzka>> getMovies() async {
  final snapshot = await _db.collection('movies').get();
```

```
  return snapshot.docs
    .map((doc) => MovieModelAzka.fromFirestore(doc))
    .toList();
}
```

3.1.3 3. Collection: `bookings`

Menyimpan transaksi booking tiket.

Fields: user_id, movie_id, movie_title, seats, total_price, booking_date, qr_data

```
Future<void> createBooking(BookingModelAzka booking) async {
  // 1. Simpan booking
  await _db.collection('bookings')
    .doc(booking.bookingId)
    .set(booking.toFirestore());

  // 2. Update booked_seats di movies
  final movieRef = _db.collection('movies').doc(booking.movieId);
  final movieDoc = await movieRef.get();

  if (movieDoc.exists) {
    final currentSeats = List<String>.from(
      movieDoc.data()!['booked_seats'] ?? []);
    final updated = [...currentSeats, ...booking.seats];
    final uniqueSeats = updated.toSet().toList();

    await movieRef.update({'booked_seats': uniqueSeats});
  }
}
```

3.2 Firebase Configuration

Konfigurasi multi-platform (Android, iOS, Web, Windows, macOS):

```
class DefaultFirebaseOptions {
  static FirebaseOptions get currentPlatform {
    if (kIsWeb) return web;
    switch (defaultTargetPlatform) {
      case TargetPlatform.android: return android;
      case TargetPlatform.iOS: return ios;
      default: throw UnsupportedError('Platform not supported');
    }
  }
}

// Inisialisasi
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  runApp(const CineBookingApp());
}
```

3.3 Service Layer: FirebaseServiceAzka

Centralized service untuk semua operasi backend.

3.3.1 Authentication Methods

```
class FirebaseServiceAzka {
  final FirebaseAuth _auth = FirebaseAuth.instance;
  final FirebaseFirestore _db = FirebaseFirestore.instance;

  Future<User?> loginUser({required String email, required String password})
async {
    final cred = await _auth.signInWithEmailAndPassword(
      email: email.trim(), password: password.trim());
    return cred.user;
  }

  Future<void> logoutUser() => _auth.signOut();

  User? getCurrentUser() => _auth.currentUser;
}
```

3.3.2 Booking Methods dengan Fallback

```
Future<List<BookingModelAzka>> getUserBookings(String userId) async {
  try {
    final snapshot = await _db.collection('bookings')
      .where('user_id', isEqualTo: userId)
      .orderBy('booking_date', descending: true)
      .get();
    return snapshot.docs.map((doc) =>
      BookingModelAzka.fromFirestore(doc)).toList();
  } catch (e) {
    // Fallback jika index belum dibuat
    if (e.toString().contains('index')) {
      final allSnapshot = await _db.collection('bookings').get();
      return allSnapshot.docs
        .map((doc) => BookingModelAzka.fromFirestore(doc))
        .where((b) => b.userId == userId)
        .toList()
        ..sort((a, b) => b.bookingDate.compareTo(a.bookingDate));
    }
    return [];
  }
}
```

3.4 Error Handling

```
try {
  await createBooking(booking);
} catch (e) {
  if (e.toString().contains('network')) {
    throw 'Tidak ada koneksi internet';
  } else if (e.toString().contains('permission')) {
    throw 'Akses ditolak';
```

```
  } else {
    throw 'Gagal membuat booking';
  }
}
```

3.5 Firestore Security Rules

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /users/{userId} {
      allow read: if request.auth != null;
      allow write: if request.auth.uid == userId;
    }
    match /movies/{movieId} {
      allow read: if true;
      allow write: if false; // Admin only
    }
    match /bookings/{bookingId} {
      allow read: if request.auth.uid == resource.data.user_id;
      allow create: if request.auth != null;
    }
  }
}
```

3.6 Kesimpulan

Backend CineBooking menggunakan arsitektur clean dengan 3 collection utama (users, movies, bookings), model classes terpisah, service layer centralized, error handling robust dengan fallback, dan dukungan multi-platform.

BAB 4
Implementasi Antarmuka & Autentikasi

4.1 Implementasi UI oleh Dian

4.1.1 Home Screen

HomeScreenDian menampilkan daftar film dalam grid 2 kolom dengan fitur: header user info, pull-to-refresh, loading/error state, dan floating button.

Grid Movies dengan Empty State:

```
Widget _buildMoviesGrid() {
  if (_movies.isEmpty) {
    return Center(
      child: Column(
        children: [
          Icon(Icons.movie_outlined, size: 60),
          Text('No movies available'),
          ElevatedButton(onPressed: _refreshData, child: Text('Refresh')),
        ],
      ),
    );
  }

  return GridView.builder(
    gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
      crossAxisCount: 2,
      crossAxisSpacing: 12,
      mainAxisSpacing: 12,
      childAspectRatio: 0.62,
    ),
    itemCount: _movies.length,
    itemBuilder: (_, index) => Hero(
      tag: 'movie-${_movies[index].movieId}',
      child: MovieCardDian(
        movie: _movies[index],
        onTap: () => _navigateToDetail(_movies[index].movieId),
      ),
    ),
  );
}
```

4.1.2 Movie Card Widget

Card kompak dengan poster, rating, duration, price, dan badge availability (Available/Almost Full/Sold Out).

```
class MovieCardDian extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final isSoldOut = movie.availableSeats == 0;
    final isAlmostFull = movie.availableSeats > 0 &&
```

```
                        movie.availableSeats <= 5;

    return GestureDetector(
      onTap: isSoldOut ? null : onTap,
      child: Container(
        decoration: BoxDecoration(
          borderRadius: BorderRadius.circular(8),
          color: AppColors.netflixGrey,
        ),
        child: Column(
          children: [
            // Poster dengan loading/error handling
            ClipRRect(
              child: Image.network(movie.posterUrl,
                height: 140, fit: BoxFit.cover),
            ),
            // Info: title, rating, duration, price, badge
            Container(
              padding: EdgeInsets.all(10),
              child: Column(
                children: [
                  Text(movie.title, maxLines: 2),
                  Row(children: [
                    Icon(Icons.star, size: 12),
                    Text('${movie.rating}'),
                  ]),
                  Row(children: [
                    Text('Rp ${movie.basePrice}'),
                    Container(
                      decoration: BoxDecoration(
                        color: isSoldOut ? AppColors.seatSold
                          : isAlmostFull ? AppColors.warningOrange
                          : AppColors.successGreen,
                      ),
                      child: Text(isSoldOut ? 'SOLD OUT'
                        : '${movie.availableSeats} left'),
                    ),
                  ]),
                ],
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

4.1.3 Movie Detail Screen

Tampilan detail dengan poster full-width, gradient overlay, dan informasi lengkap.

```
Widget _buildPosterSection() {
  return Stack(
    children: [
      Image.network(_movie.posterUrl, height: 400, fit: BoxFit.cover),
      Container(
        height: 400,
        decoration: BoxDecoration(
          gradient: LinearGradient(
            colors: [Colors.transparent, Colors.black.withOpacity(0.7)],
          ),
        ),
      ),
      Positioned(
        bottom: 20, left: 20, right: 20,
        child: Column(
          children: [
            Text(_movie.title, style: TextStyle(fontSize: 32)),
            Row(
              children: [
                Icon(Icons.star), Text('${_movie.rating}'),
                Icon(Icons.timer), Text('${_movie.duration} min'),
                Container(
                  color: AppColors.netflixRed,
                  child: Text('Rp ${_movie.basePrice}'),
                ),
              ],
            ),
          ],
        ),
      ),
    ],
  );
}
```

4.2 Implementasi Auth oleh Vina

4.2.1 Login Screen dengan Validasi

LoginScreenAzka dengan validasi lengkap untuk login dan register.

Validasi Form:

```
Future<void> _submitForm(BuildContext context) async {
  // Basic validation
  if (_emailController.text.isEmpty || _passwordController.text.isEmpty) {
    setState(() => _error = 'Please fill all fields');
    return;
  }

  // Register validation
  if (!_isLogin) {
    if (_passwordController.text != _confirmPasswordController.text) {
      setState(() => _error = 'Passwords do not match');
```

```
      return;
    }

    if (!CalculationServiceNadif.validateStudentEmail(_emailController.text))
{
      setState(() => _error =
        'Only student emails allowed (@student.univ.ac.id)');
      return;
    }

    if (_usernameController.text.length < 3) {
      setState(() => _error = 'Username must be at least 3 characters');
      return;
    }

    if (_passwordController.text.length < 6) {
      setState(() => _error = 'Password must be at least 6 characters');
      return;
    }
  }

  // Process
  setState(() => _isLoading = true);
  try {
    final service = Provider.of<FirebaseServiceAzka>(context, listen: false);

    if (_isLogin) {
      await service.loginUser(
        email: _emailController.text.trim(),
        password: _passwordController.text.trim(),
      );
    } else {
      await service.registerUser(
        email: _emailController.text.trim(),
        password: _passwordController.text.trim(),
        username: _usernameController.text.trim(),
      );
      await service.loginUser(
        email: _emailController.text.trim(),
        password: _passwordController.text.trim(),
      );
    }
  } catch (e) {
    setState(() => _error = e.toString());
  } finally {
    setState(() => _isLoading = false);
  }
}
```

UI dengan Error Display:

```
// Error message
if (_error.isNotEmpty)
  Container(
    decoration: BoxDecoration(color: Colors.red[900]),
    child: Row(
      children: [
        Icon(Icons.error),
        Text(_error),
      ],
    ),
  ),

// Submit button
ElevatedButton(
  onPressed: _isLoading ? null : () => _submitForm(context),
  child: _isLoading
    ? CircularProgressIndicator()
    : Text(_isLogin ? 'Sign In' : 'Sign Up'),
)
```

4.2.2 Navigation System

Auth Wrapper:

```
class AuthWrapper extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final service = Provider.of<FirebaseServiceAzka>(context);
    final user = service.getCurrentUser();

    return user != null ? HomeScreenDian() : LoginScreenAzka();
  }
}
```

Splash Screen:

```
class SplashScreen extends StatefulWidget {
  @override
  State<SplashScreen> createState() => _SplashScreenState();
}

class _SplashScreenState extends State<SplashScreen> {
  @override
  void initState() {
    super.initState();
    Future.delayed(Duration(seconds: 2), () {
      Navigator.pushReplacement(
        context,
        MaterialPageRoute(builder: (_) => AuthWrapper()),
      );
    });
  }

  @override
```

```
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.black,
      body: Center(
        child: Column(
          children: [
            Icon(Icons.movie, size: 80, color: Color(0xFFE50914)),
            Text('CINEBOOKING', style: TextStyle(fontSize: 32)),
            CircularProgressIndicator(),
          ],
        ),
      ),
    );
  }
}
```

4.3 Theme & Constants

Color palette Netflix-inspired untuk konsistensi UI:

```
class AppColors {
  static const netflixRed = Color(0xFFE50914);
  static const netflixDark = Color(0xFF141414);
  static const netflixBlack = Color(0xFF000000);
  static const netflixGrey = Color(0xFF2D2D2D);

  static const seatAvailable = Color(0xFF404040);
  static const seatSelected = Color(0xFF2196F3);
  static const seatSold = Color(0xFFFF44336);
}
```

4.4 Kesimpulan

UI implementation mencakup: HomeScreen dengan grid responsif, MovieCard dengan badge status, MovieDetail dengan hero animation, LoginScreen dengan validasi lengkap (email mahasiswa, password 6+ karakter, username 3+ karakter), dan navigation system dengan AuthWrapper dan SplashScreen. Semua menggunakan theme konsisten Netflix-inspired.

BAB 5
State Management & Transaction Logic

Bagian ini dikerjakan oleh Nadif, mencakup state management dengan Provider dan business logic untuk booking.

5.1 Setup Provider

```
class CineBookingApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MultiProvider(
      providers: [
        ChangeNotifierProvider(create: (_) => BookingControllerNadif()),
        Provider(create: (_) => FirebaseServiceAzka()),
      ],
      child: MaterialApp(...),
    );
  }
}
```

5.2 Booking Controller

BookingControllerNadif mengelola state dan logic booking.

5.2.1 State Variables & Methods

```
class BookingControllerNadif extends ChangeNotifier {
  List<String> _selectedSeats = [];
  List<BookingModelAzka> _userBookings = [];
  List<String> _bookedSeats = [];
  String? _currentUserId;
  bool _isLoading = false;
  String? _error;

  // Getters
  bool get hasSelectedSeats => _selectedSeats.isNotEmpty;

  // Seat selection
  void toggleSeat(String seatId) {
    if (!_bookedSeats.contains(seatId)) {
      _selectedSeats.contains(seatId)
        ? _selectedSeats.remove(seatId)
        : _selectedSeats.add(seatId);
      notifyListeners();
    } else {
      _error = 'Kursi $seatId sudah dipesan';
      notifyListeners();
    }
  }
}
```

5.2.2 Create Booking dengan Validasi

```
Future<String?> createBooking({
  required String movieId,
  required String movieTitle,
  required int basePrice,
}) async {
  // Validasi
  if (_currentUserId == null) throw 'Silakan login terlebih dahulu';
  if (_selectedSeats.isEmpty) throw 'Pilih minimal satu kursi';

  for (final seat in _selectedSeats) {
    if (_bookedSeats.contains(seat)) {
      throw 'Kursi $seat sudah dipesan orang lain';
    }
  }

  _isLoading = true;
  notifyListeners();

  try {
    final bookingId = _uuid.v4();
    final totalPrice = calculateTotalPrice(movieTitle, basePrice);
    final qrData = CalculationServiceNadif.generateQRData(...);

    final booking = BookingModelAzka(
      bookingId: bookingId,
      userId: _currentUserId!,
      movieId: movieId,
      movieTitle: movieTitle,
      seats: List.from(_selectedSeats),
      totalPrice: totalPrice,
      bookingDate: DateTime.now(),
      qrData: qrData,
    );

    await _firebaseService.createBooking(booking);
    await Future.wait([loadUserBookings(), loadBookedSeats(movieId)]);

    _selectedSeats.clear();
    notifyListeners();
    return bookingId;
  } catch (e) {
    _error = e.toString().contains('network')
      ? 'Tidak ada koneksi internet'
      : 'Gagal membuat booking';
    notifyListeners();
    rethrow;
  } finally {
    _isLoading = false;
    notifyListeners();
```

```
  }
}
```

5.3 Calculation Service

CalculationServiceNadif menangani semua perhitungan business logic.

5.3.1 Logika Perhitungan

```
// Parse seat number
static int parseSeatNumber(String seatCode) {
  return int.tryParse(seatCode) ?? 0;
}

static bool isEvenSeat(String seatCode) {
  return parseSeatNumber(seatCode) % 2 == 0;
}

// Harga kursi dengan diskon genap 10%
static int calculateSeatPrice(String seatCode, int basePrice) {
  if (isEvenSeat(seatCode)) {
    return (basePrice * 0.9).toInt(); // Diskon 10%
  }
  return basePrice;
}

// Pajak judul >10 karakter
static int calculateTitleTax(String title, int seatCount) {
  return title.length > 10
    ? AppConstants.titleTax * seatCount  // Rp 2.500/kursi
    : 0;
}

// Total harga
static int calculateTotalPrice({
  required List<String> seats,
  required String movieTitle,
  required int basePrice,
}) {
  if (seats.isEmpty) return 0;

  int total = 0;
  for (final seat in seats) {
    total += calculateSeatPrice(seat, basePrice);
  }
  total += calculateTitleTax(movieTitle, seats.length);

  return total;
}
```

5.3.2 Contoh Perhitungan

Film: “The Dark Knight Returns” (24 karakter), Base: Rp 50.000, Kursi: [1,2,3,4]

```
Kursi 1 (ganjil): Rp 50.000
Kursi 2 (genap):  Rp 45.000 (diskon 10%)
Kursi 3 (ganjil): Rp 50.000
Kursi 4 (genap):  Rp 45.000 (diskon 10%)
Subtotal:         Rp 190.000
Pajak (4×2.500):  Rp 10.000
TOTAL:            Rp 200.000
```

5.3.3 Price Breakdown

```
static String generatePriceBreakdown({
  required String movieTitle,
  required List<String> seats,
  required int basePrice,
}) {
  final evenSeats = seats.where((s) => isEvenSeat(s)).length;
  final oddSeats = seats.length - evenSeats;
  final titleTax = calculateTitleTax(movieTitle, seats.length);
  final discount = (basePrice * 0.1 * evenSeats).toInt();

  return '''
=== PRICE BREAKDOWN ===
Base: Rp $basePrice x ${seats.length}
Even Seats ($evenSeats): -10% (Rp $discount)
Odd Seats ($oddSeats): Regular
Title Tax: ${titleTax > 0 ? '+Rp $titleTax' : 'No tax'}
TOTAL: Rp ${calculateTotalPrice(...)}
''';
}
```

5.3.4 QR Code Generation

```
static String generateQRData({
  required String bookingId,
  required String movieTitle,
  required List<String> seats,
  required int totalPrice,
  required DateTime bookingDate,
  required String movieId,
  required String userId,
}) {
  return '''
🎬🎞️ CINEBOOKING TICKET 🎬🎞️
==============================
MOVIE: $movieTitle
SEATS: ${seats.join(', ')}
TOTAL: Rp $totalPrice
DATE: ${DateFormat('dd/MM/yyyy HH:mm').format(bookingDate)}
BOOKING ID: $bookingId
==============================
⚠ PERMANENT - Non-refundable
SCAN FOR THEATER ENTRY
''';
}
```

5.3.5 Helper Functions

```
static String formatDate(DateTime date) =>
  DateFormat('dd MMM yyyy').format(date);

static String formatDateTime(DateTime date) =>
  DateFormat('dd/MM/yyyy HH:mm').format(date);

static bool validateStudentEmail(String email) =>
  email.endsWith('@student.univ.ac.id');

static String getSeatSummary(List<String> seats) {
  final evenSeats = seats.where((s) => isEvenSeat(s)).toList();
  final oddSeats = seats.where((s) => !isEvenSeat(s)).toList();

  String summary = '';
  if (evenSeats.isNotEmpty)
    summary += 'Even: ${evenSeats.join(', ')} (-10%)\n';
  if (oddSeats.isNotEmpty)
    summary += 'Odd: ${oddSeats.join(', ')} (regular)';
  return summary.trim();
}
```

5.4 Penggunaan di UI

5.4.1 Consumer Pattern

```
Consumer<BookingControllerNadif>(
  builder: (context, controller, _) {
    return Column(
      children: [
        Text('Selected: ${controller.selectedSeats.join(', ')}'),
        Text('Total: Rp ${controller.calculateTotalPrice(...)}'),

        ElevatedButton(
          onPressed: controller.hasSelectedSeats && !controller.isLoading
              ? () => _confirmBooking(controller)
              : null,
          child: controller.isLoading
              ? CircularProgressIndicator()
              : Text('Confirm Booking'),
        ),

        // Error display
        if (controller.error != null)
          Container(
            child: Row(
              children: [
                Icon(Icons.error),
                Text(controller.error!),
                IconButton(
                  icon: Icon(Icons.close),
                  onPressed: controller.clearError,
```

```
                ),
              ],
            ),
          ),
        ],
      );
    },
  )
```

5.4.2 QR Dialog

```
void _showQrDialog(BookingModelAzka booking) {
  showDialog(
    context: context,
    builder: (_) => Dialog(
      child: Column(
        children: [
          Text('YOUR TICKET'),
          QrImageView(data: booking.qrData, size: 180),
          Text(booking.movieTitle),
          Text('Seats: ${booking.seats.join(', ')}'),
          Text('Total: Rp ${booking.totalPrice}'),
        ],
      ),
    ),
  );
}
```

5.5 Kesimpulan

State management dengan Provider memberikan: reactive UI (perubahan state langsung terlihat), centralized logic (semua perhitungan di CalculationService), reusable functions, testable code (logic terpisah dari UI), dan maintainable structure. Business logic unik: diskon kursi genap 10%, pajak judul >10 karakter Rp 2.500/kursi, dan QR code generation.

BAB 6
Integrasi & Version Control

6.1 Strategi Branching Git

Tim menggunakan Git Flow dengan branch terpisah untuk setiap fitur:

```
main (production)
  |-- feature/backend-setup (Azka)
  |-- feature/ui-widgets (Dian)
  |-- feature/auth-nav (Vina)
  |-- feature/transaction-logic (Nadif)
```

Commit Convention:

```
feat: add movie detail screen
fix: resolve seat selection bug
style: update color scheme
refactor: improve calculation logic
docs: add README documentation
```

6.2 Integrasi Antar Modul

6.2.1 Backend Integration (Azka → All)

FirebaseServiceAzka sebagai centralized API:

```dart
// Di HomeScreen (Dian)
final movies = await FirebaseServiceAzka().getMovies();

// Di BookingController (Nadif)
await _firebaseService.createBooking(booking);
```

6.2.2 UI Integration (Dian → All)

Component reusable dengan styling konsisten:

```dart
// MovieCard dipakai di berbagai screen
MovieCardDian(movie: movie, onTap: () => navigateToDetail(movie.movieId))

// Constants untuk konsistensi
Container(
  color: AppColors.netflixRed,
  child: Text('Title', style: AppTextStyles.movieTitle),
)
```

6.2.3 State Management (Nadif → All)

Consumer pattern di semua screen:

```dart
// Di SeatSelectionScreen (Vina)
Consumer<BookingControllerNadif>(
  builder: (context, controller, _) => Column(
    children: [
      Text('Selected: ${controller.selectedSeats.length} seats'),
      Text('Total: Rp ${controller.calculateTotalPrice(...)}'),
```

```
      ElevatedButton(
        onPressed: controller.hasSelectedSeats ? () => _confirm() : null,
        child: Text('Confirm'),
      ),
    ],
  ),
)

// Di HomeScreen (Dian) - statistik user
Consumer<BookingControllerNadif>(
  builder: (context, controller, _) {
    final total = controller.userBookings.fold(0, (sum, b) => sum +
b.totalPrice);
    return Text('${controller.userBookings.length} bookings, Rp $total
spent');
  },
)
```

6.2.4 Navigation Integration (Vina → All)

Routing konsisten:

```
// Login → Home
Navigator.pushReplacement(context,
  MaterialPageRoute(builder: (_) => HomeScreenDian()));

// Home → Detail → Seat Selection → Profile
Navigator.push(context, MaterialPageRoute(...));
```

6.3 Error Handling Strategy

6.3.1 Network & Loading States

```
try {
  _isLoading = true;
  notifyListeners();
  await firebaseService.createBooking(booking);
} catch (e) {
  _error = e.toString().contains('network')
    ? 'Tidak ada koneksi internet'
    : 'Gagal membuat booking';
} finally {
  _isLoading = false;
  notifyListeners();
}

// Di UI
if (_isLoading) CircularProgressIndicator()
else if (_error != null) ErrorWidget(error: _error)
else ContentWidget()
```

6.3.2 Empty State

```
if (_movies.isEmpty) {
  return Center(
```

```
    child: Column(
      children: [
        Icon(Icons.movie_outlined, size: 60),
        Text('No movies available'),
        ElevatedButton(onPressed: _refreshData, child: Text('Refresh')),
      ],
    ),
  );
}
```

6.4 Testing Checklist

Authentication: Register (valid/invalid email), Login (benar/salah), Logout

Movie Browsing: Load data, detail film, hero animation, refresh, empty state

Seat Selection: Pilih available, reject booked, toggle, hitung harga real-time

Booking: Validasi, QR generation, save Firebase, update booked_seats

Calculation: Diskon genap 10%, pajak judul >10 char, total akurat

6.5 Performance Optimization

```
// 1. Lazy Loading
ListView.builder(
  itemCount: bookings.length,
  itemBuilder: (context, index) => BookingCard(booking: bookings[index]),
)

// 2. Image Caching (otomatis di Flutter)
Image.network(movie.posterUrl, fit: BoxFit.cover)

// 3. Batch Updates
Future<void> refreshAll() async {
  _isLoading = true;
  notifyListeners(); // 1x

  await Future.wait([loadMovies(), loadBookings(), loadProfile()]);

  _isLoading = false;
  notifyListeners(); // 1x
}
```

6.6 Deployment

Platform Support: Android, iOS, Web, Windows, macOS

Build Commands:

```
flutter build apk --release      # Android
flutter build ios --release      # iOS
flutter build web --release      # Web
```

6.7 Collaboration Workflow

```
# Feature Development
git checkout -b feature/auth-nav
git add .
git commit -m "feat: add login screen"
git push origin feature/auth-nav

# Merge
git checkout main
git merge feature/auth-nav
git push origin main
```

6.8 Documentation

```
# CineBooking App
Flutter + Firebase cinema booking app

## Features
- Authentication, Movie browsing, Seat selection
- QR Code tickets, Booking history

## Setup
1. Clone repo
2. flutter pub get
3. Configure Firebase
4. flutter run

## Team
Azka (Backend), Dian (UI), Vina (Auth), Nadif (Logic)
```

6.9 Kesimpulan

Integrasi sukses dengan: centralized FirebaseService API, consistent UI components dan styling, reactive state dengan Provider, robust navigation pattern, comprehensive error handling (network/loading/empty), performance optimization (lazy loading, caching, batch updates), dan structured Git workflow.

BAB 7
Pengujian & Penutup

7.1 Handling Data

- **Link:** [https://github.com/DanishNaisyila/cine-booking-team4.git]